MICHAEL A. JACOBS (CA SBN 111664)
MJacobs@mofo.com
MATTHEW I. KREEGER (CA SBN 153793)
MKreeger@mofo.com
MORRISON & FOERSTER LLP
425 Market Street
San Francisco, California 94105-2482
Telephone: (415) 268-7000
Facsimile: (415) 268-7522

KYLE W.K. MOONEY (*Pro Hac Vice*)
KMooney@mofo.com
ERIC W. LIN (*Pro Hac Vice*)
ELin@mofo.com
MICHAEL J. DESTEFANO (*Pro Hac Vice*)
MDeStefano@mofo.com
MORRISON & FOERSTER LLP
250 West 55th Street
New York, New York 10019-9601
Telephone: (212) 468-8000
Facsimile: (212) 468-7900

ROSE S. LEE (CA SBN 294658)
RoseLee@mofo.com
MORRISON & FOERSTER LLP
707 Wilshire Boulevard, Suite 6000
Los Angeles, California 90017-3543
Telephone: (213) 892-5200
Facsimile: (213) 892-5454

Attorneys for Defendant
PALO ALTO NETWORKS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

FINJAN LLC,

Plaintiff,

v.

PALO ALTO NETWORKS, INC.,

Defendant.

Case No. 3:14-CV-04908-JD

DEFENDANT PALO ALTO NETWORKS, INC.'S WRITTEN TECHNOLOGY SYNOPSIS

Judge: Honorable James Donato

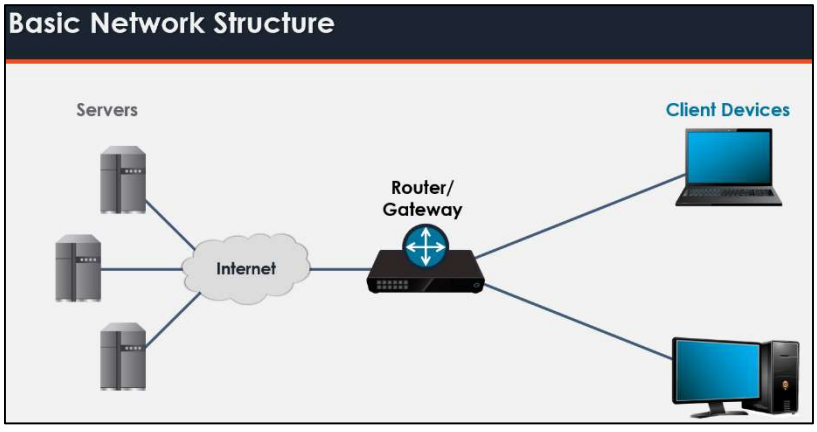
Pursuant to the Amended Scheduling Order (ECF No. 228) and Paragraph 8 of the Standing Order for Claim Construction in Patent Cases, Defendant Palo Alto Networks, Inc. submits this written technology synopsis describing background information about the technology relevant to the four patents asserted by Plaintiff Finjan LLC: U.S. Patent Nos. 8,141,154 ("'154 Patent"); 7,647,633 ("'633 Patent"); 8,225,408 ("'408 Patent"); and 7,418,731 ("'731 Patent"). These patents generally relate to network security and protecting user computers from harmful incoming content from the Internet, *e.g.*, malware or viruses.

I. NETWORK DEVICES AND SECURITY BACKGROUND

Basic Network Structure. A computer network comprises multiple computers (*e.g.*, web servers, PCs, or laptops) that communicate with each other. The computers in a computer network can work together to have greater functionality and computational power than a single computer. Organizations use computer networks to perform various tasks, store information, and manage and control a diverse range of infrastructure.

Computers in a computer network are often connected to other computers on the network or other computer networks through connection points known as routers. The computers communicate with each other using a communication protocol. The Internet can be considered a "network of networks" that connects billions of public and non-public networks.

Many Internet communications follow the "Client-Server" model. In this model, the computer requesting data is called a "client" device and the computer responding to requests is called a "server." For example, when a user downloads a file using a browser, the user's computer acts as a client and the source of the download is a server. Because malware and viruses can be transmitted over the Internet, many antivirus software programs are designed to run on client devices and scan data when it is received from a server. Communication transmitted over the Internet,



however, is not transferred directly from the server to the client. Internet communication is routed from the source (*i.e.*, server) to the destination (*i.e.*, requesting client device) across intermediary devices called routers. Non-public corporate and home networks have at least one router called a "gateway" that marks the boundary between them and the Internet. The gateway marking the boundary between the non-public network and the Internet processes all traffic in and out of the network. As a result, gateways have become useful as security devices.

Data Sent via Packets. Data or information is sent over a network via small units known as "packets." A network packet is analogous to a physical package sent via traditional mail to an apartment building. The physical package needs a shipping label that specifies the address and apartment number. The shipping label address would be the equivalent of an IP address, which is a unique number that identifies a device on a network, and the apartment number would be the equivalent of a port number, which identifies which program or service on that device will be used to exchange information over a network. A packet consists of (1) a header and (2) a payload. Packet headers are the computer's version of shipping labels that include routing information (*i.e.*, IP addresses and port numbers), and the payload is the packet's contents. Files sent over a network are typically broken down into multiple packets.

Network Security. The purpose of network security tools is to protect computers from harmful programs by either blocking downloadable malware or preventing its execution on user computers. Security gateways and firewalls are frequently used for such purposes. As discussed, gateways are devices that sit on the edge of a computer network that all traffic entering or exiting the network passes through. Security gateways can employ a firewall, which is a network security system that monitors and controls incoming and outgoing network traffic based on defined security rules. A security gateway employing a firewall can allow, block, or transform traffic based on the results of the security inspection.

Network security has evolved significantly since the 1990s when commercial firewalls first appeared. There are now many different techniques to protect computers from malware. Early firewalls were simple systems focused on inspecting and filtering incoming network packets. If the traffic did not match the packet filter's rules, the firewall would drop or reject the

packet.

The early firewalls from the 1990s evolved into "stateful" filters that kept track of connections between computers and could determine if a packet was part of a new or existing connection, which made filtering easier. In response to a rising need in the 2000s to monitor software applications (*e.g.*, web browsers and email programs), Unified Threat Management ("UTM"), the second generation of firewalls, added gateway antivirus, intrusion detection, and prevention capabilities by bolting them on to existing stateful firewalls. UTM brought disparate network security technologies together into a single application, but there were gaps in security and low performance due in part to complex policy management of the disparate technologies. Next Generation Firewalls ("NGFW"), the third generation of firewalls, addressed these issues by providing the capabilities of traditional firewalls while simplifying policy management and offering increased flexibility. NGFW features include enhanced application visibility and control, and user identity awareness and protection. NGFWs are often centrally managed and allow for more robust and sophisticated security decisions because, for example, they can differentiate network traffic based on the application or user associated with the network traffic.

Defense Methodologies. One of the earliest approaches in network security to defeat malware is the "scanner." The basic concept behind a scanner is to analyze a given unit of data and compare it to known patterns and sequences of malware. Scanning at the firewall could potentially intercept malware in a download, and scanning at an email gateway could block a virus before a user ever opens the email.

"Signature scanning" was an early technique in antivirus software that involves scanning portions of files for a particular pattern of characters or instructions (*i.e.*, a "signature") found in known malware. Signature scanning requires a database of known patterns found in (and preferably only in) malware. The scanning software uses that database to compare patterns of known malware with the pattern of the incoming data or information. If the scanned data or information has the same sequence of bytes of known malware, the scanner (or another program combined with the scanner) does not allow the program to run.

Another early approach to antivirus software is "hash matching," which in contrast to

signature scanning, involves collecting an entire file rather than only portions of a file. Hash matching is a process that uses a "hash function" (a function that produces a fixed-size alphanumeric value to represent a whole file). In hash matching, the hash function takes a file that can be downloaded to another computer (*i.e.*, a "downloadable") and applies the hash function to generate the "hash ID." After the hash ID is generated for the downloadable, that hash ID is checked against a database of hash IDs of known malware. If there is a match, then there is an indication that the file is malicious and the scanner (or another program combined with the scanner) will block the file.

"Static scanning" or "static analysis" is another defense methodology. Static analysis involves examining a program without actually executing it to observe virus-like behavior—in particular, a program being analyzed is compared to known viruses that have been logged in a database. If enough of the program matches what is in the database, the program is flagged as a potential threat.

Lastly, "dynamic analysis," also known as "runtime analysis," involves executing a file in a safe, simulated environment (referred to as a "sandbox") to determine whether the file is malicious. If during execution suspicious behavior is detected, an alert will be generated and the file will be blocked or flagged as a potential threat. Dynamic analysis has existed since at least the early 1990s. Dynamic analysis aids in detecting zero-day attacks, which are attacks that have never been seen before. Static analysis tools may not be able to detect some zero-day attacks because they may be unable to process malware with encrypted or obfuscated instructions. One downside of dynamic analysis, however, is that it requires a lot of processing power and could significantly slow down a computer system if it is performed on every incoming file.

II. CONCEPTS OF THE '154 AND '633 PATENTS

At a high level, the '154 and '633 Patents pertain to runtime inspection and sandboxing. To discuss the relevant concepts in more detail, it is important to understand the basics of source code. A computer program is a sequence of instructions that tells computer processors what to do. The program is written by a programmer in source code using a text editor. Source code is in a format that the programmer can read and write. Once the program is complete, the source code

is "compiled" into an executable file that is in machine code. Machine code consists of the zeros and ones that a computer reads and executes.

A source code function call is a sequence of program instructions that perform a specific task. For example, a function "`max(int num1, int num2)`" could be defined by a programmer to return the larger of two integers. A function call can typically be recognized by a function name, such as "`max`," followed by parentheses. The parentheses are for inputs to the function call. In the example of `max(7,10)`, 7 and 10 are the inputs to the function "`max`," and the output of the function would be 10, which is the maximum of the two integers passed as input.

Sometimes while a function call itself may seem innocuous, the input to a function can be malicious, making malicious behavior harder to detect. For example, with the function `document.write("DeleteAll")`, the function may need to be run with the input to determine whether the behavior is indeed malicious. One goal of network security is to prevent potentially malicious code from harming a computer network. The network security industry has developed many ways to do it.

One approach proposed by the '154 Patent is to examine a function's input at runtime at a security computer away from the user. Taking a look at Figure 2 of the '154 Patent (annotated on the right), there are three functional components: (1) gateway computer (yellow); (2) client computer (green); and (3) security computer (blue). Starting with the gateway (yellow), the gateway receives content containing an original function with suspicious input (*e.g.*, "`Function(input)`"). The gateway modifies the content by substituting the original function with a substitute function (*e.g.*, "`Substitute_function(input,*)`"). The modified content is then sent to the client computer (green). The client computer receives the modified content and runs the modified content until it reaches the substitute function. The substitute function then tells the client

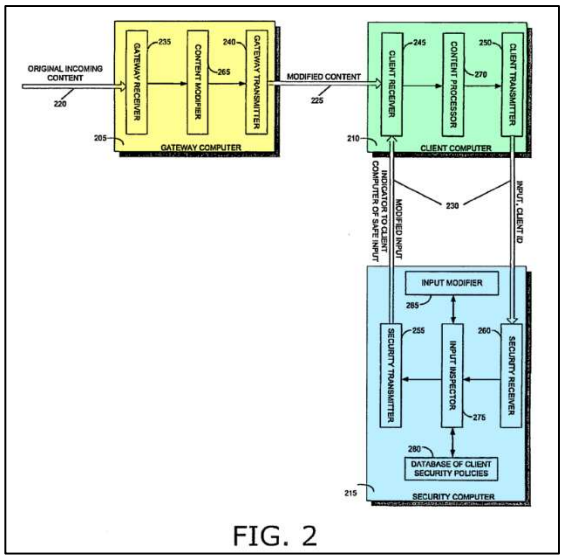


FIG. 2

computer to send the suspicious input to the security computer (blue) for analysis. The security computer receives the suspicious input from the client computer and analyzes the input. After analysis, the security computer indicates to the client computer if the input may be safely provided to the function to run.

The '633 Patent addresses a similar concept: sandboxing. Sandboxing is a security mechanism for separately executing programs that may contain malicious content to analyze their behaviors. If malicious behaviors are detected by the sandbox as a program executes, then the content is blocked. Figure 1b of the '633 Patent (reproduced on the right) shows an example of a protection-initiating "server" or re-communicator" 140a that is capable of transferring or "re-communicating" downloadable information to user device 145. Server 141b includes a protection engine ("PE") 142a that includes an information monitor for monitoring information received by the server and a code detection engine for determining whether the received information includes executable code. The PE also includes a packaging engine. If the information received by the server is determined to be downloadable or executable, the packaging engine creates a sandboxed package including mobile protection code ("MPC") and downloadable protection policies and sends the package to a downloadable-destination (*i.e.*, the user device 145) for execution by the user device 145. The MPC in the sandboxed package has different capabilities. It can cause predetermined malicious operations to be monitored and intercepted or initiate executing the downloadable in a protective sandbox.

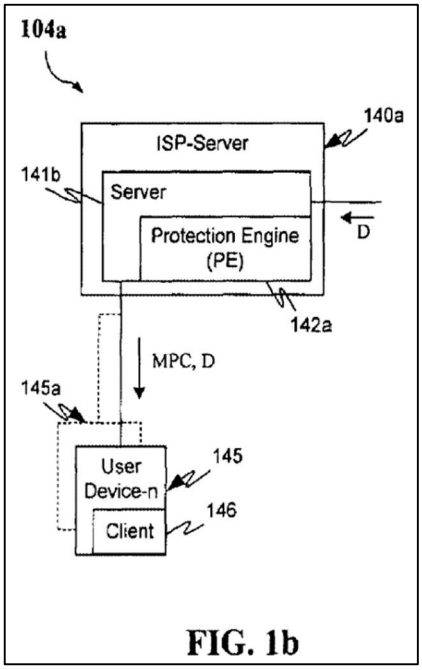
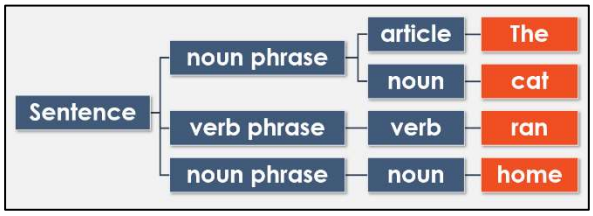


FIG. 1b

III. CONCEPTS OF THE '408 PATENT

The '408 Patent pertains to a dynamic scanner and parser that operate on source code written in a programming language. As discussed above, source code consists of program statements created by a programmer with a text editor. Source code has not yet been compiled into an executable file, meaning that source code is still in a format that a programmer can read

and write. Parsing breaks down a data stream written in a programming language into its component parts called "tokens" so that the meaning of the stream can be understood. The parser then organizes these tokens based on the grammatical rules of that programming language.



A parse tree graphically depicts the parsed stream. The image to the right helps illustrate the concept of parsing, but using an English sentence rather than a programming language. In this example the English sentence is parsed into nouns, verbs, and articles, which would be analogous to tokens in a programming language.

Figure 2 of the '408 Patent (annotated on the right) shows an example of a dynamic scanner and parser. The tokenizer (yellow) breaks down an input data stream into individual tokens. The parser (green) places those tokens in a parse tree in accordance with certain parser rules. The analyzer (blue) then analyzes the parse tree for patterns of malicious content. The dynamic scanner and parser of the '408 patent only scans and analyzes source code that a programmer can read or write. It does not scan or analyze machine code compiled into zeros and ones.

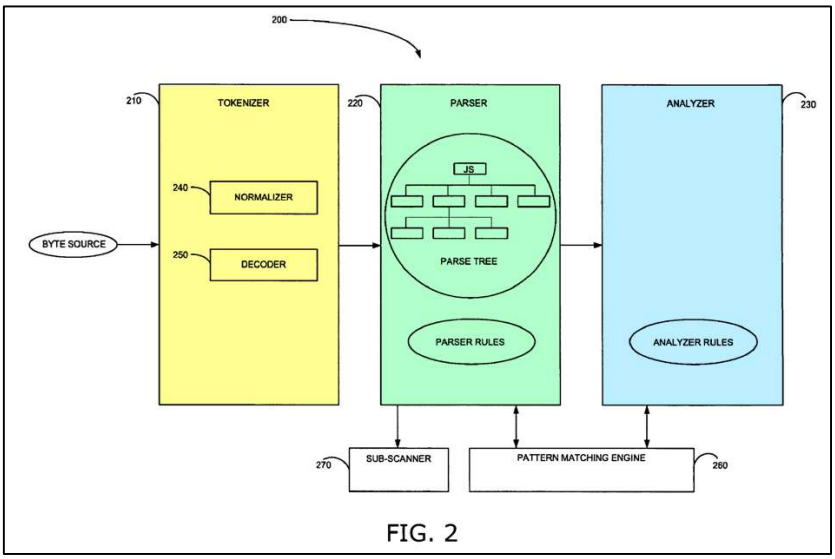
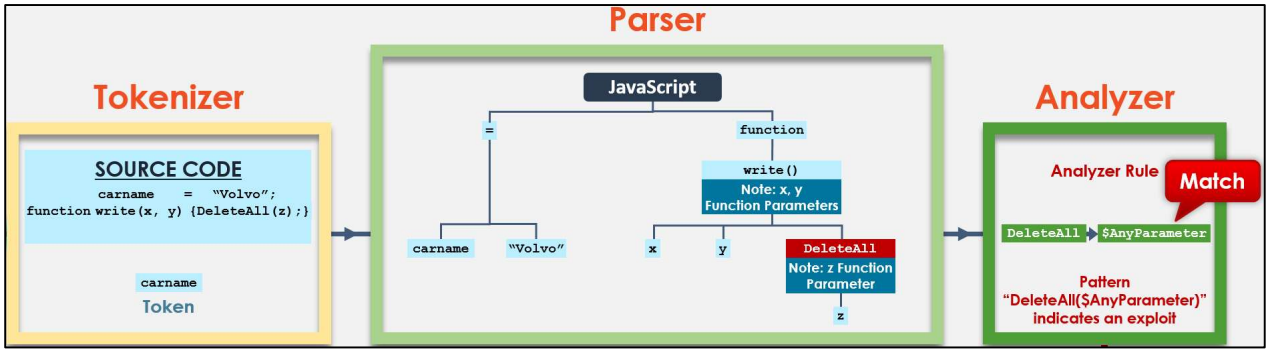


FIG. 2

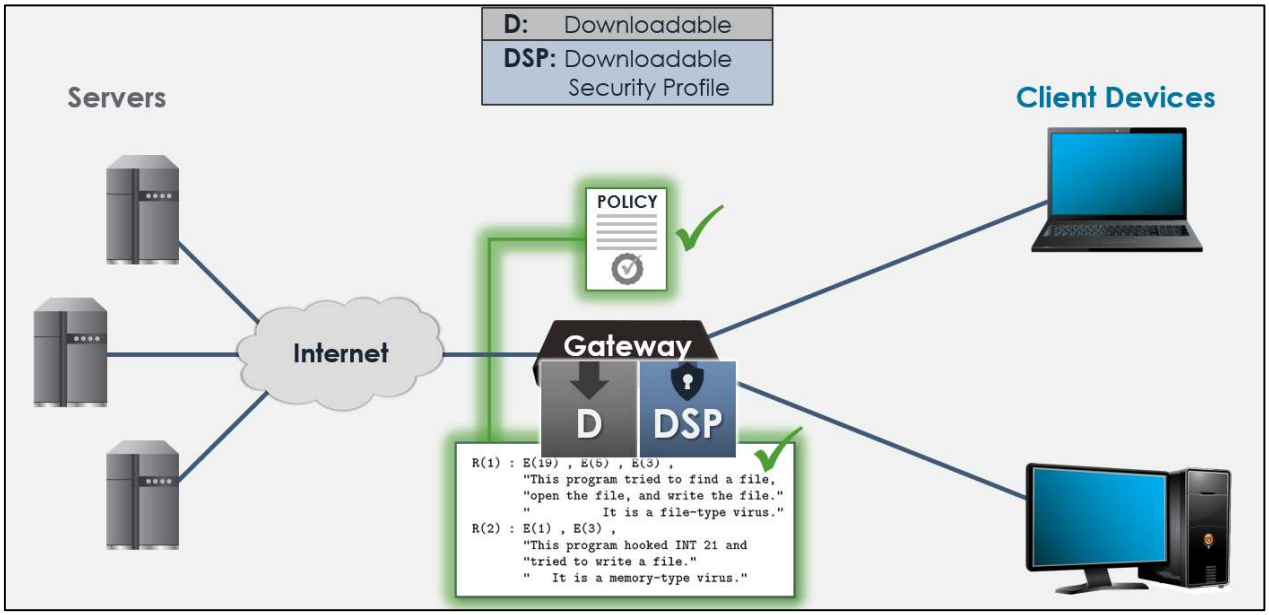
Below is an example of dynamic scanning in real time. In the first step, the tokenizer



reads the source code until it recognizes complete tokens. In the example, the first complete token is "carname." Each time the tokenizer identifies a token, it passes the token on to the parser. As the tokens are passed to the parser, the parser builds the parse tree, which contains a node for each token (*e.g.*, "`carname`," "`Volvo`," "`function`," "`write()`," etc.). The parser builds the parse tree in real time while the system is still receiving an incoming stream of program code. The parser then passes nodes to the analyzer as the parser builds the parse tree. The analyzer uses analyzer rules to determine if a node matches a known malicious exploit. In the example above, the analyzer rule is the statement that the pattern "`DeleteAll($AnyParameter)`" indicates an exploit. When nodes "`DeleteAll`" and "`z`" reach the analyzer, the analyzer finds a match for a potential exploit and can take the action designated by the analyzer rule, such as sending a notification that a threat has been identified.

IV. CONCEPTS OF THE '731 PATENT

The '731 Patent pertains to security profiles and caching. As shown in the example below, a client device requests a downloadable (D) from an internet server. When the downloadable arrives at the gateway, the gateway generates a downloadable security profile (DSP) that identifies the computer commands that the downloadable is programmed to perform (*e.g.*, "This program tried to find a file, open the file, and write the file."). The gateway then



attaches the DSP to the downloadable and makes the downloadable with DSP available for download. The decision whether to transmit the downloadable to the requesting client device is based on a security policy from the client device. The security policy may indicate suspicious operations that are to be blocked from the client device. The DSP is compared against the security policy and if there is no indication of suspicious operations to be blocked from the client device, then the downloadable is allowed through to the requesting client device.

Using "caching" can save time and resources when retrieving content that had previously been retrieved and analyzed. A local cache temporarily stores data to increase subsequent retrieval speeds and save bandwidth. In the example above, if Computer B in a network requests File Q from a web server, that file will be downloaded and stored in cache memory. If Computer C also requests File Q at a later time, it can obtain the file from the cache instead of requesting it from the web server all over again.

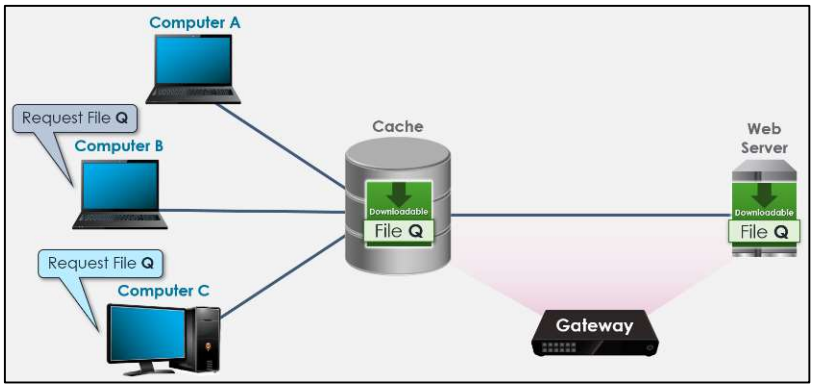


Figure 1 in the '731 Patent (reproduced below) shows an example of generating a security profile and using caches. A gateway computer 110 intervenes between requests from client devices in an intranet 120 and responses from internet servers 133, 135, and 137. The gateway computer 110 includes a code scanner 140 for scanning incoming web pages. Scanner 140 analyzes each file it scans to determine the nature of computer operations that the file is programmed to perform and derives a security profile summarizing the potentially malicious computer operations.

When a client computer requests a web page from a server computer, the request is first transmitted to gateway computer 110, which checks whether the web page is already within web cache 160. The gateway determines which files to scan, derives security profiles for the scanned files, and caches the

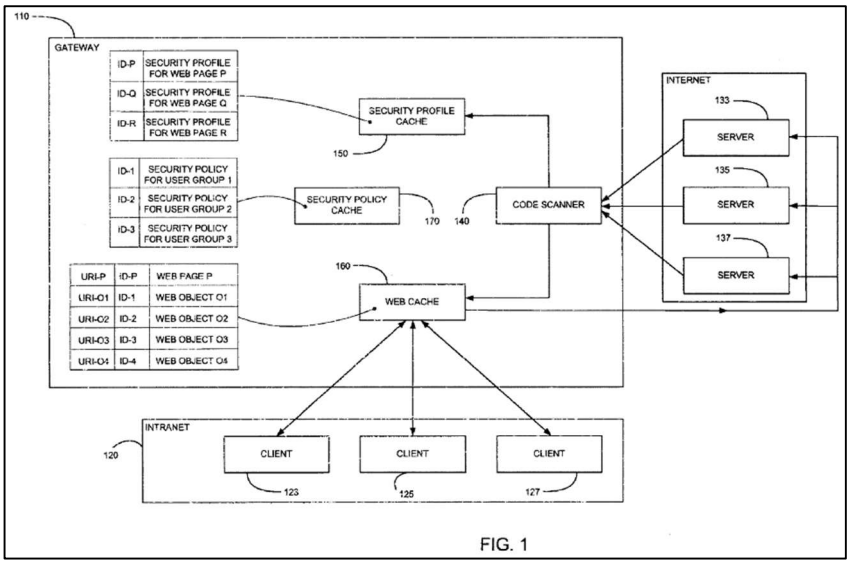
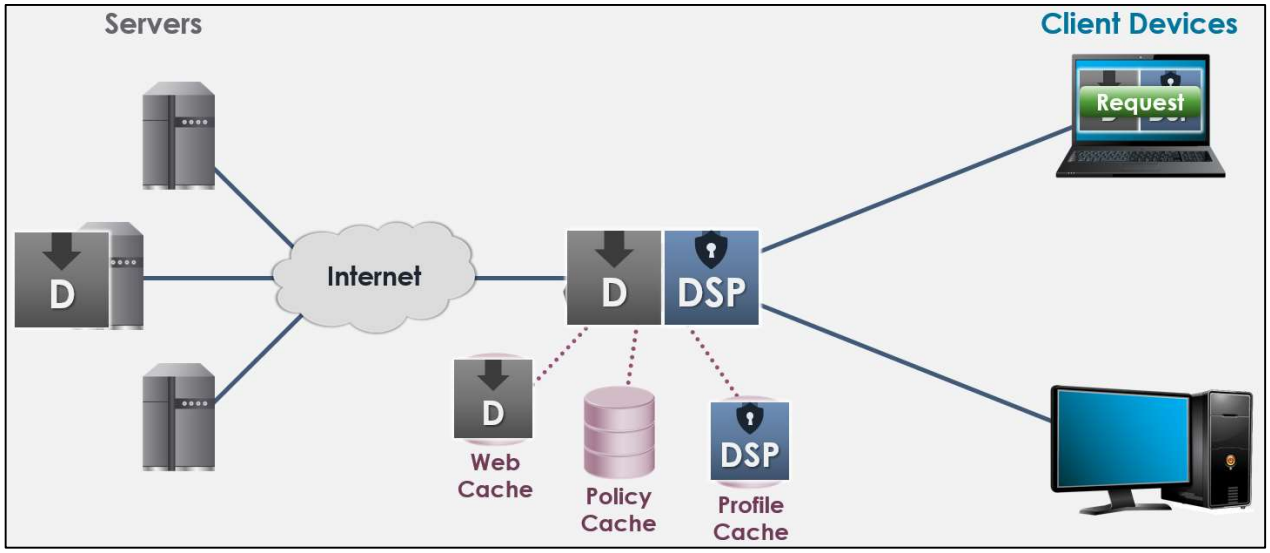


FIG. 1

security profiles in the security profile cache 150. The decision whether or not to transmit the web page to the requesting client computer is based on a security policy for the client computer. The security policy indicates suspicious operations that are to be blocked from the client computer. Security policies are stored within a security policy cache 170. The security profile cache 150, web cache 160, and security profile cache 170 must be managed in order to be kept current. When cache memory is full and new items arrive for storage, older items from the cache are purged.

When a client device requests a downloadable from a web server, caches can be created for the downloadable content requested from the web (*i.e.*, web cache), for the security policies of



the client device (*i.e.*, policy cache), and for the security profiles generated for the downloadable (*i.e.*, profile cache) (shown in the image above). If a second client device requests the same downloadable at a later time, the gateway receives the request and checks the web cache to see if the cache already has the downloadable. If the web cache already has the requested content, the gateway will retrieve the DSP for that downloadable, which includes a list of computer commands that the downloadable is programmed to perform. The gateway then retrieves the security policy for the intranet client computer from the security policy cache and compares the listed commands of the DSP against the security policy to determine whether to allow or block the downloadable. If all the computer commands are permitted under the security policy, the downloadable is transmitted to the client device.

Dated: May 5, 2023

MORRISON & FOERSTER LLP

By: */s/ Matthew I. Kreeger*
Matthew I. Kreeger

Attorneys for Defendant
PALO ALTO NETWORKS, INC.